




Mission Management Computer Software for RLV-TD

C. R. Manju¹ · Josna Susan Joy¹ · L. Vidya¹  · I. Sheenarani¹ · C. N. Sruthy¹ ·
P. C. Viswanathan¹ · Sudin Dinesh¹ · L. Jayalekshmy¹ · Kesavabrahmaji Karuturi² ·
E. Sheema¹ · S. Syamala² · S. Manju Unnikrishnan¹ · S. Akbar Ali¹ ·
R. Paramasivam¹ · D. S. Sheela¹ · A. Abdul Shukkoor² · V. R. Lalithambika¹ ·
T. Mookiah¹

Received: 18 May 2017 / Accepted: 16 October 2017
© The Institution of Engineers (India) 2017

Abstract The Mission Management Computer (MMC) software is responsible for the autonomous navigation, sequencing, guidance and control of the Re-usable Launch Vehicle (RLV), through lift-off, ascent, coasting, re-entry, controlled descent and splashdown. A hard real-time system has been designed for handling the mission requirements in an integrated manner and for meeting the stringent timing constraints. Redundancy management and fault-tolerance techniques are also built into the system, in order to achieve a successful mission even in presence of component failures. This paper describes the functions and features of the components of the MMC software which has accomplished the successful RLV-Technology Demonstrator mission.

Keywords Mission Management Computer · Real-time operating system · Sequencing

Introduction

The RLV Navigation Guidance and Control (NGC) configuration consists of Mission Management Computer (MMC) and input-output modules. They are interconnected using highly deterministic communication bus in dual-redundant configuration. The MMC acts as Bus Controller (BC) in the bus and all other modules are configured as Remote Terminals (RT). The software in the MMCs processes real-time inputs from Inertial Navigation Systems

(INS) and other input RTs and posts the control outputs to control actuators through output RTs. The different components of MMC software are Real-Time Executive (REX), Navigation, Sequencing, Guidance and Control modules.

NGC Configuration

The NGC configuration in RLV is shown in Fig. 1. MMC and input-output RTs are provided in dual redundancy to achieve required fault tolerance; they are also cross-strapped to tolerate two non identical failures. The prime modules are connected as paired RTs in the communication bus of each MMC and the redundant modules are connected as cross strapped RTs.

MMC Software Components

REX

REX software is a custom-built Real Time Operating System (RTOS) for MMC which is based on Intel i960 32bit processor [1] with in-built floating point unit. REX shall support pre-flight and flight mode of operations of MMC. REX is designed and developed for space, time and cost efficiency, incorporating all functional and reliability requirements of MMC. It is developed in high-level and assembly languages [2].

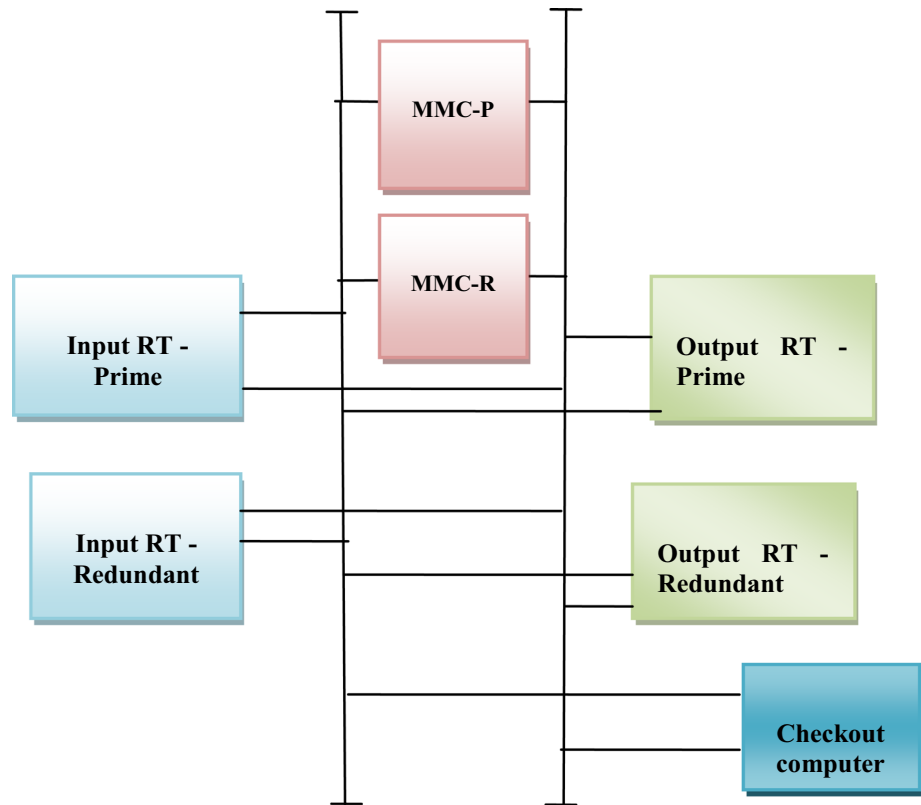
REX Requirements

In pre-flight mode, also known as monitor mode, MMC shall be a RT in the 1553B bus which is controlled by a

✉ L. Vidya
srividhya.laila@gmail.com; l_vidya@vssc.gov.in

¹ Vikram Sarabhai Space Centre, Thiruvananthapuram, India

² ISRO Inertial Systems Unit, Thiruvananthapuram, India

Fig. 1 System configuration

checkout computer. The checkout computer also issues power 'on' and hardware resets to MMC. Monitor mode enables the loading and reading of the program or the data into the memory, the execution of self test routines onboard to evaluate health of processor, the communication links, the timer and the memory resources in MMC. After these extensive health evaluations, the MMC is taken to flight mode of operation.

In flight mode, the navigation and control shall be scheduled at every micro cycle (10 ms periodicity) in ascent phase and at every minor cycle (20 ms periodicity) afterwards. This dynamic reconfiguration of the scheduling is implemented based on Real-Time Decision (RTD) event corresponding to ascent-to-descent transition of RLV during its flight trajectory. Sequencing software and periodic self check of MMC processor is done at every minor cycle throughout the mission. Navigation, guidance and temperature compensation of the ceramic servo accelerometers is executed at every major cycle (500 ms periodicity). The smallest period tasks shall be given highest priority.

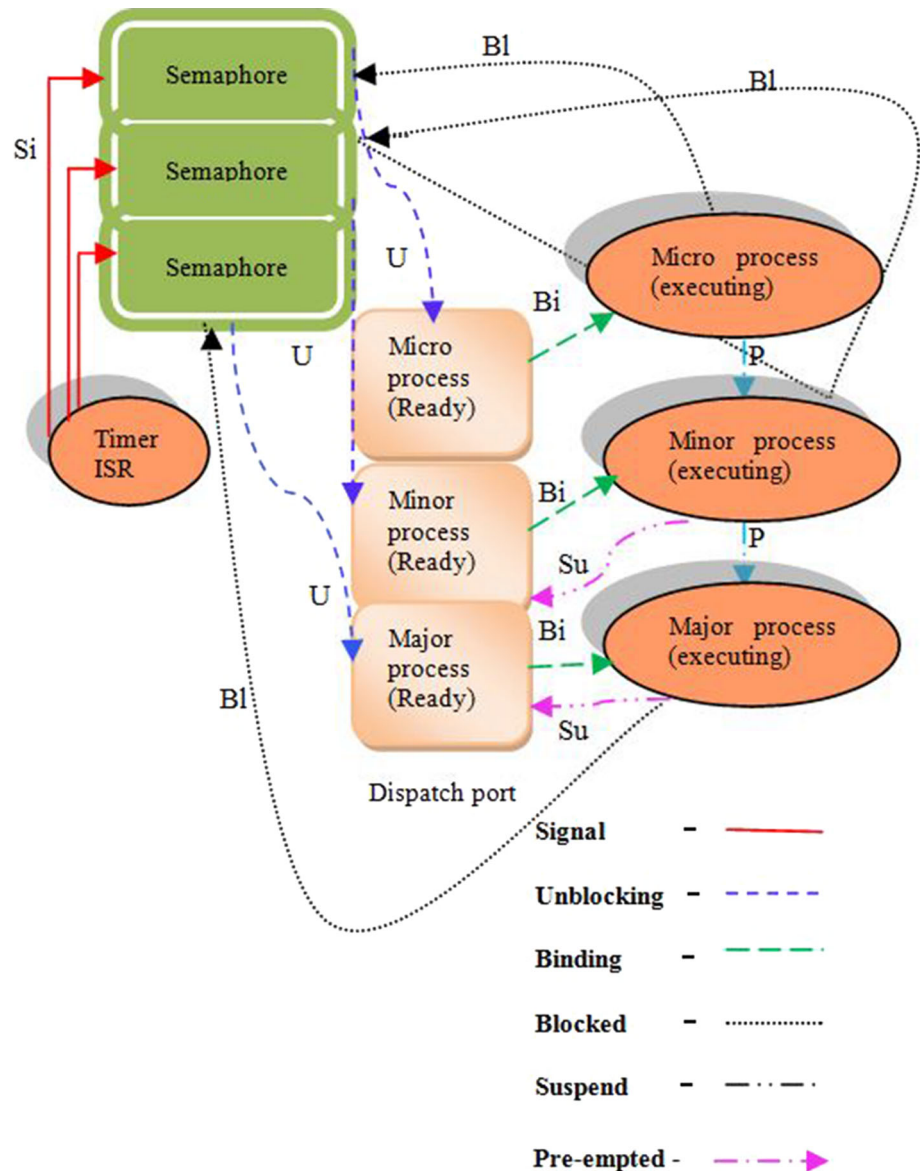
Scheduling of MMC software modules shall be done according to their periodicity, priority and precedence requirements. Prior to scheduling, REX shall collect inputs from input RTs and do error handling to utilize fully cross-strapped dual redundancy. The output of the MMC software modules, its sequencing and the control information, shall be posted to output RTs along with MMC health to

enable them to utilize redundancy in MMC. All the above functional requirements shall be completed within a transport delay of 10 ± 1 ms in ascent and 20 ± 2 ms in descent phase of the mission. A master-slave inter-processor synchronization scheme among the MMCs and the other processing nodes shall also be provided by the REX to accomplish the functional requirements.

The Scheduler Algorithm

The scheduler algorithm is depicted in Fig. 2.

Initially all the three processes will be blocked at their respective semaphore ports. The timer in MMC is programmed to generate interruptions at the periods of the micro cycle process. The timer Interrupt Service Routine (ISR), will derive boundaries for each of these processes and generate signals to their respective semaphore ports. On the reception of signals, the processes will be unblocked from semaphore and will be queued in the dispatch port defined for the processor. The scheduling of these processes from the dispatch port takes place according to their assigned priority. Once the process is completed, it will be again blocked at the semaphore port. Provision for process pre-emption is also provided, that is, a higher priority task can suspend the execution of lower priority task.

Fig. 2 State transition of processes

The processes can have four different states namely executing, interrupted, ready and blocked. The state of the tasks inside each process is also defined in the task table. They can have four different states namely dormant, ready, running and completed.

Error Handling Requirements of REX

REX shall periodically check the processor execution environment, its memory, the timer and the communication bus interface at every minor cycle. The error condition will be indicated to the downstream output RTs by the means of MMC Health word. MMC health will be made “not normal” depending on the severity of error condition either immediately (Category-1) or delayed (Category-2), after confirmation of the recurrence for ‘n’ consecutive cycles.

This enables to detect and isolate a faulty MMC and reconfigure to redundant healthy MMC.

Processor Related The processor self checks task exercises the used instructions like data transfer, logical, arithmetic, control flow and also exercise, all internal registers and the selected RAM locations. This error comes in category-2 to take care of transient failures in MMC. The arithmetic control word of the micro, the minor and the major cycle processes shall be verified at the respective boundaries for any arithmetic error conditions that have occurred like over flow, division by zero etc and also, at every boundary, it will be ensured that no scheduled tasks crossed their deadline. In case of any processor fault the corresponding fault handler shall be invoked. All the earlier error conditions fit into Category-1. In case of processor

bus failure no communication will be initiated on the communication bus and hence down-stream RTs shall reconfigure to take data from a valid bus.

Memory Related Single Bit Error Correction and Double Bit Error Detection of processor RAM (SECDED), TMR check on real time counters, RGTM and XOR checksum of EEPROM code area are also grouped in Category-1.

Timer Related If the timer drifts beyond the range, then synchronization of redundant MMC with prime will be disabled. If the timer fails to generate interrupts, then no bus communication shall be initiated. The down-stream RTs shall reconfigure itself to take data from other MMC.

Communication Bus Interface Related Communication protocol validation failure of all functional messages in the bus will be monitored to identify the bus interface failure. Read back verification of communication interface memory will be done to identify RAM failures. These errors are also grouped in Category-2 to take care of the transient failures in the interface chip.

Comparison with Existing Launch Vehicle REX

The existing RTOS for expendable satellite launch vehicles requires the software execution in two different periods: the minor and the major. The RLV demands three levels of periodicity. The communication bus utilization is also double compared to the existing satellite launch vehicles. The following additional factors are considered in RLV RTOS.

The Input–output management tasks are executed as the highest priority task to ensure predictable message validation and posting. The message validation and message initialization are done when the bus load is null. As the bus load is high, this factor was considered while designing the communication protocol. Different communication protocols are implemented for the ascent and the descent phases of the mission.

The control task is to be scheduled as a micro cycle task. It requires the output of the minor cycle navigation and guidance tasks. These minor cycle tasks might not have been completed within the first micro cycle itself. Hence in the second micro cycle, control task is to be executed as a minor cycle task instead of micro cycle task. This specific requirement is also handled effectively with the rate monotonic scheduler [3] implemented for RLV.

Navigation Software

Navigation system for RLV, needs a complex navigation and air data estimation algorithms, based on data from INS, GPS, Radar Altimeter, atmospheric and wind data, to meet

the requirements of various phases of mission. Multi sensor integrated navigation using Kalman filter and other advanced filtering algorithms is employed to meet the complex navigation requirements. To meet the redundancy requirements, a new variant of Redundant Strapdown INS (RESINS) with skewing of accelerometers in yaw-roll plane is used. Rate Gyro Package (RGP) and Ceramic Servo Accelerometer Package (CSAP) are used to meet the control requirements during ascent and descent phase of the vehicle, respectively. High Resolution Data Acquisition System (HDAS) is newly developed for RLV to acquire high resolution (16-bit) rate and acceleration data from RESINS. The overall functional block schematic of navigation system for RLV is shown in Fig. 3.

Navigation Software Requirements

The basic requirements of navigation, include acquisition of sensor data through the Navigation Interface Module, the error handling and redundancy management of data from various systems, the sensor data processing and generate navigation solution required for guidance, the control and the sequencing. For RLV based on navigation requirements, the entire flight trajectory can be partitioned to three different phases ascent, descent ($\text{Mach} > 2$) and descent ($\text{Mach} < 2$).

Navigation requirements during the ascent phase include, generation of quaternion and rate feedback to DAP at 10 ms periodicity, altitude to guidance in 20 ms periodicity used for open-loop steering, inertial velocity magnitude for DAP in 20 ms used for gain scheduling.

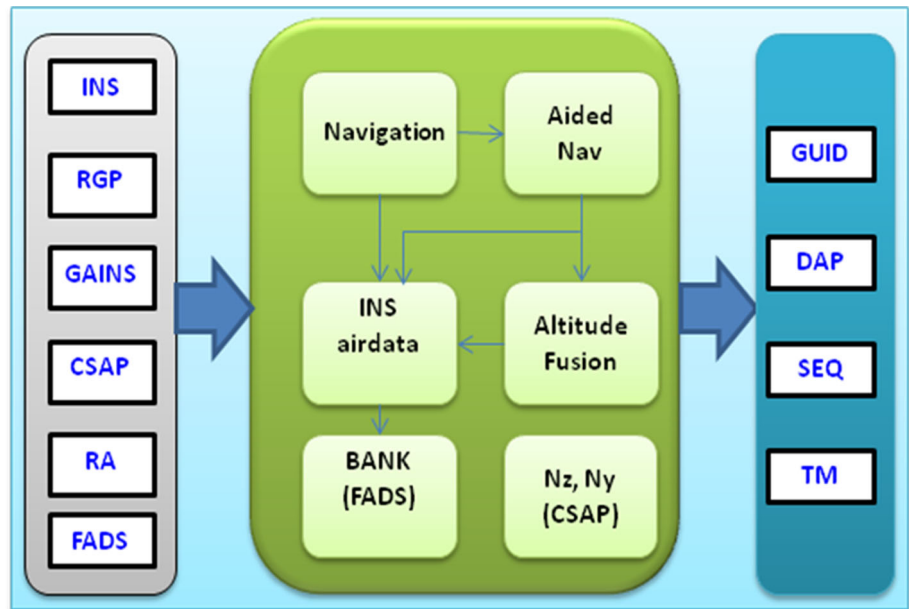
Navigation requirements for descent phase ($\text{Mach} > 2$) include, generation of aerodynamic angles, rate feedback for DAP in 20 ms periodicity, ground relative mach to guidance for open-loop steering, mach and dynamic pressure to DAP for gain scheduling.

Navigation requirements for descent phase ($\text{Mach} < 2$) include, generation of normal and lateral acceleration and rate feedback to DAP in 20 ms periodicity, altitude, position and velocity for guidance, mach and dynamic pressure to DAP for gain scheduling.

To meet the accuracy requirements of RLV, GPS aiding is used in active loop to correct the INS solution. To improve on the altitude accuracy during the final terminal phase, Radar Altimeter data is used for correcting the navigation altitude below 2 km. An observer based algorithm with gain scheduling is employed for this purpose.

Airdata estimation based on navigation data, ground loaded nominal atmospheric data and day of launch measured wind data is designed and implemented to meet the guidance and DAP requirements during the descent phase. The accuracy of aerodynamic angles computed is sensitive to wind variations at lower mach numbers, hence normal

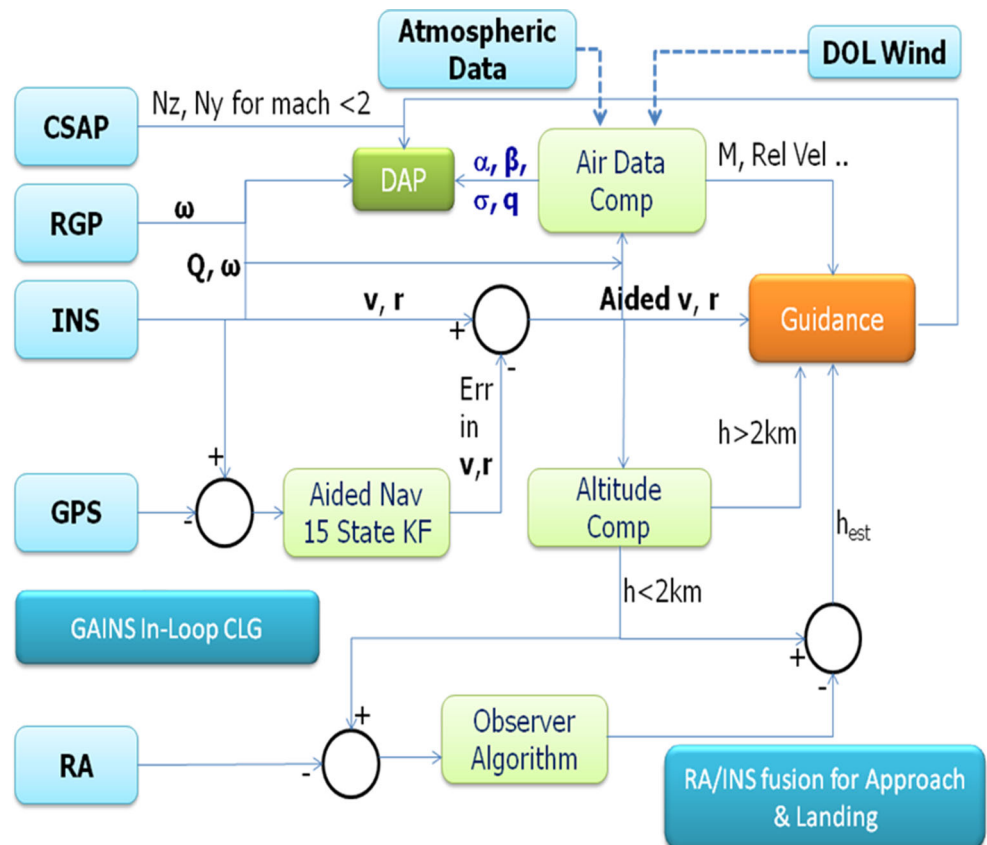
Fig. 3 Functional block schematic of navigation system for RLV



and lateral accelerations are used for control for Mach < 2. The overall computational block schematic of the navigation software is shown in Fig. 4.

GPS Aided INS System For GPS aiding, a 15 state Kalman filter is employed to estimate the errors in INS state vector and correct in forward path. The model for the Kalman filter is the error dynamics of the INS. The input measurement to the Kalman filter is the difference (INS-

Fig. 4 Computational block schematic of navigation software



GPS). The state variables of the filter are selected after considering the mission trajectory, velocity and position accuracy requirements. The algorithm is designed to handle spurious GPS samples with the capability of self-restart in case of continuous GPS data outage or any communication errors.

For ensuring accurate estimate of the INS position and the velocity errors, robustness of the extended Kalman filter used in GAINS system is ensured by providing various levels of integrity checks and error handling logics. The GPS data validity checks are done at input level before start of the Kalman filter. The data is validated by ensuring GPS data integrity, GPS receiver (hardware) check, availability of fresh GPS solution, availability of solution in 3D mode and GPS PDOP checks. To ensure accuracy of the solution, GPS-INS data synchronization is done using minor cycle INS velocity.

The estimated errors obtained from GAINS will be fed to navigation for correcting INS position and velocity. The aided position and the velocity will be further used for computation of air data parameters and for the altitude computation.

Navigation Interface Module Interface module is the sensor data acquisition and processor interface module. It acquires rate and acceleration data from gyros and accelerometers. The major design considerations include.

- Redundancy: two identical NIM are present in the INS, each with two independent 1553B links.
- Modularity and Testability: NIM can be separately tested in tests beds by simulating the sensor inputs.
- Data Acquisition Hardware: NIM supports 48 analog data acquisition channels acquired through ADC and 16 pulse accumulators to acquire digital pulses from VFC. NIM also provides the relay enable signals to switch on the sensors at ground.
- Data Acquisition Periodicity: NIM performs data acquisition at micro cycle periodicity in ascent and minor cycle periodicity in descent phase. Major cycle acquisition is done for monitoring parameters.

Synchronization with MMC NIM (prime and redundant) are configured to synchronize to the closest MMC. NIM provides the health status of the data acquisition electronics and software to MMC for real time decision. The status word in NIM provides information for the analog and digital channel failure, the processor failure and the 1553B communication message timeout/synchronization status.

Comparison with Existing Launch Vehicle Navigation

RLV being a complex vehicle with a winged body structure sitting on a slender solid motor, the navigation requirements are much higher complex compared to the existing launch vehicles. The major changes in the navigation requirements with respect to the existing launch vehicles of ISRO include.

- Quaternion integration in 10 ms periodicity, for control during the ascent phase.
- Position and velocity update in 20 ms periodicity with GPS aiding in closed-loop.
- New failure detection and isolation scheme for roll-yaw plane acceleration channels.
- Airdata parameters estimation during the descent phase using ground loaded atmospheric and wind data.
- Altitude aiding using radar altimeter, during the terminal phase (altitude < 2 km).

Sequencing Software

During the flight regime of RLV, a series of operations and commands have to be executed in a predefined sequence. It is the function of the sequencing software residing in the MMC to execute this sequence of commands for RLV mission.

Sequencing Software Requirements

The sequencing commands are executed with respect to certain events which are sensed in real time. These events are the Real-Time Decision (RTD) events which are detected based on trigger conditions like, Last Minute Plug (LMP) status, navigation parameters like total inertial velocity, thrust axis velocity increments, ground relative mach number and altitude flag. The RTDs are related to critical events like stage tail off, control scheme switch over, meeting target conditions etc.

The redundancy of the NGC system is used to ensure that the RTDs are detected in both the MMCs. If RTD is detected in one MMC, it is forced in the other MMC by RTD synchronisation.

The sequencing software issues software commands to the other software residing within the MMC like the navigation, the guidance, the control and the REX software. The software flag contains information regarding different phases of the flight. The sequencing software also issues the hardware commands like staging commands to the hardware sequencer which carries out down-the-line execution.

Error Handling in Sequencing Software

Error free execution of sequencing commands in the launch vehicle is achieved by the exhaustive fault detection and isolation [4] schemes implemented in the sequencing software.

The RTDs are detected within a time window in order to prevent false detection. In case the trigger condition is not satisfied within the window, the RTD is forced at the window out. In order to prevent false detection of RTD, a consistency check is done on the inputs. The continuously varying navigation inputs is checked against the threshold value for three consecutive minor cycles to ensure that the RTD is not detected due to noisy signal.

In order to prevent the generation of wrong sequencing commands due to memory corruption, the entire sequencing software is implemented as two independent software streams. Each stream of the sequencing software contains the required inputs, independent processing modules and generates the outputs. This ensures that any memory corruption in one stream does not affect the other stream. The final outputs of both the streams are checked for data integrity. The final sequencing outputs are passed by sequencing software only if the outputs of both the streams agree.

The sequencing software issues software flags and flag complements to the application software. The integrity of the flags are verified by the different software before usage. Similarly the hardware commands and command complements are posted to the hardware sequencer, which carries out the integrity check of the hardware commands.

The non-execution of sequencing commands will lead to mission failure. The redundancy of the NGC system is made use of to ensure that the sequencing commands are executed either from the prime or redundant chain.

Onboard Guidance Software

Main objective of guidance software in RLV-TD is to generate optimal steering commands in every minor cycle to guide the vehicle from an initial condition to the desired landing point meeting various path constraints. Guidance algorithms and commands generated during re-entry are entirely different from that of launch vehicles since RLV-TD is a lifting body and vehicle control is through aerodynamic surfaces during descent phase.

Guidance Software Requirements

Guidance software takes information about the vehicle states from navigation, the sequencing information from sequencer, other system parameters and flags from REX and posts the computed steering commands to Digital Auto

Pilot (DAP) and required flag information to REX. The block schematic of guidance software is given in Fig. 5.

For guidance algorithm design, the mission is divided into the following phases as shown in Fig. 6.

(i) ascent phase, (ii) orientation phase, (iii) initial entry phase, and (iv) approach and landing phase.

During the ascent phase, the vehicle is steered using commanded quaternion derived from an altitude based day of launch wind biased steering profile as in first stage of conventional launch vehicles. After booster separation, it is commanded through aerodynamic angles such as commanded angle of attack, side slip angle and bank angle. This is continued till a pre-defined Mach value is reached and below which vehicle is commanded through commanded normal acceleration, lateral acceleration and bank angle till touchdown. Closed-loop guidance command computations to steer the vehicle to the desired landing point are initiated below a pre-defined Mach number only.

In case of over or under performance of the vehicle, guidance software shall have a provision to retarget the landing points. This feature shall be invoked through initialization. Guidance software shall also have a provision to dynamically shift the landing point based on the vehicle performance. In case of severe over or under performance of the vehicle, the guidance software shall have a provision to avoid the closed-loop computations and generate commands using an alternate scheme called Safe Mode Guidance (SMG) scheme.

Error Handling in Guidance Software

The integrity of the sequencing flag inputs is verified before usage to detect any memory failure. In the case of any error, the software enables REX to make health of MMC as not normal for reconfiguration to healthy MMC. Input range checking is done on trigonometric or square root library routines and software proceeds with the clamped value if the range exceeds. Software detects

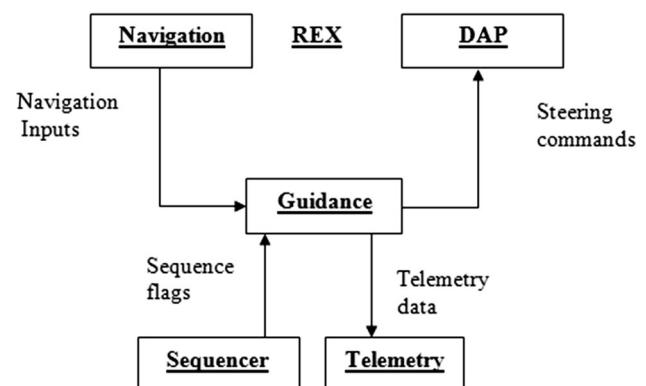


Fig. 5 Computational block schematic of guidance software

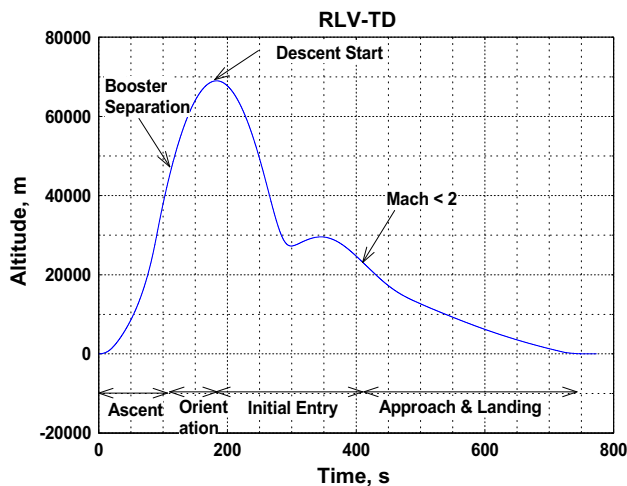


Fig. 6 RLV mission profile

division by zero error and enters to SMG in case of error. All array indexes are bounded and counters are clamped by design.

Comparison with Existing Launch Vehicles

As far as guidance is concerned, the algorithm schemes in ascent phase are similar to that of conventional expendable launch vehicles. Guidance schemes and commands during re-entry are entirely different from that of launch vehicles since the vehicle is an aerodynamic lifting body and the control is through aerodynamic control surfaces during descent phase. It takes various inputs from other subsystems and produces different outputs during different regimes of the flight. Hence the number of modules that can be reused from existing launch vehicle software is very minimal.

In launch vehicle, most of the events required by guidance will be provided by sequencer based on related RTDs and these will occur in a pre-defined sequence only. Scheduling of the guidance activities is not much difficult. In re-entry mission, for the technology demonstrator phase, most of the events required by guidance have to be detected by itself and based on the performance of the vehicle, they may not occur in the expected order. This fact is to be considered while designing the internal sequencing and scheduling of activities in re-entry mission.

Error recovery schemes for guidance computational errors are different since it is a re-entry mission.

Onboard DAP Software

The Digital Autopilot (DAP) software computes the control commands for stabilizing the attitude of the vehicle and tracking the guidance commands. The DAP software issues

control commands during the RLV ascent and descent phases computed using the sensor data, the sequencing data and the guidance parameters in every minor cycle. One minor cycle is 10 ms during ascent phase and 20 ms during descent phase. DAP processing is based on input parameters which may vary over time and other fixed initialization parameters which can be configured.

The system operates in four phases and DAP software is required to compute control commands during the four different phases. The system states will be represented by the sequence Flag input and the current phase of the system. Depending on the control parameters, processing is done for the appropriate phase. Processing time for one minor cycle is expected to be < 2 ms for ascent-phase and 4 ms for descent-phase in an i960 processor. The entire processing is done sequentially.

In the ascent phase, attitude commands (quaternions) are tracked by the DAP whereas in the descent phase up to Mach-2, Alpha-Beta-Sigma commands are tracked. Below Mach-2, since the accuracy of input Alpha and Beta are not sufficient, acceleration commands and Sigma are tracked.

In the ascent phase until the booster FINS become effective, Secondary Injection Thrust Vector Control System (SITVC) is used to control the pitch and yaw channels whereas Reaction Control System (RCS) of RLV-TD is used for roll control. Further in the ascent phase booster FINS are used for attitude control up to booster separation. Thereafter the RLV-TD is controlled by elevons (pitch and roll control) and rudders (yaw control) and RCS is in backup mode of operation. In the initial phase of the descent, when the rudder control is not effective, RCS is used for control in the yaw channel.

The DAP algorithm computes the body error using the inputs from the navigation and guidance software. After the filtering of inputs at the appropriate places, the control law is executed to compute the delta error function. Subsequently, the control commands are generated by the DAP depending on the phase of the RLV flight.

DAP Software Requirements

The DAP Software receives the attitude or acceleration commands from the guidance software; The current vehicle parameters (attitude quaternion, acceleration, alpha, beta, sigma, body rates, mach number, dynamic pressure and inertial velocity) are received from the navigation software and the vehicle sequencing events from the sequencing software. The output DAP commands are routed to the control power plants by a real-time executive software.

The major features of RLV-TD DAP software are given here:

1D/2D Gain Scheduling One-dimensional gain scheduling is based on inertial velocity in ascend phase and Mach number in descend phase. 2D gain scheduling is based on the Mach number and the dynamic pressure.

Trim Command Generation Vehicle trim commands are generated from the stored 2D table based on Mach and commanded angle of attack. These values are added to the final DAP commands in the descent phase.

Attitude, Rate and Forward Path Filters Provision for second order, fourth order and eighth order filters. The higher order filters are implemented as a parallel combination of second order blocks.

Filter Switching Provision for different number of filter switching in various phases. During filter switching duration, output of the current and the new filter are blended.

Rate Control Logic Limits The software has provision for applying different limits on errors in different phases. This logic helps to have a smooth capture from large initial conditions.

Command Rate Limiting This feature limits the rate of the final DAP commands to control the actuators.

Control Law Switching On switching over from one phase to another with a different control law, two control law outputs are blended for a specified duration.

Computation of Control Commands Depending upon the phase of operation, control commands to various control power plants are computed.

Salvage Schemes It has mission salvage schemes in case of accelerometer failure.

Filter Transition Logic The ascent phase of the software is executed with a periodicity of 10 ms whereas the subsequent phases are executed with a periodicity of 20 ms. The filters at the end of 10 ms phase are continued to the next phase (20 ms) and this requires a smooth filter transition logic. New filter transition logic was developed to transfer the internal states of the 10 ms filter to 20 ms filter with a minimal computational complexity.

2D Interpolation Algorithm Without Explicit Division by Zero Check The gains in the descent phase of the autopilot are computed from a pre-stored two dimensional table based on Mach number and dynamic pressure. The 2D interpolation algorithm avoids division by zero in the presence of identical corner points in Mach number or Dynamic Pressure. Gains are clamped at the boundaries if input is outside the range of the table to avoid spurious gains.

Error Handling in DAP Software

Sufficient software protection has been given to avoid the error conditions in the software such as division by zero, constraint error (if input parameter is outside the range of gain table) and non-monotonicity of parameters such as Mach number, dynamic pressure and AlphaC.

In the DAP software division by zero protection is implemented in places where there is a possibility with respect to the denominator. At one place in the software division is implemented by zero protection is the computations where dynamic pressure occurs in the denominator. Another place where protection against division by zero is needed is the computation of time offset during salvage condition in the ascent and descent phases.

The final DAP commands are limited to integer range to avoid overflow.

Salvage Plans in DAP Software RESINS accelerometer failure or CSAP failure creates a situation in which inputs will not be available to the DAP from the navigation such as the inertial velocity, the Mach number, the dynamic pressure, and the accelerations (Nz and Ny). Mission salvage plans are a part of the DAP software to handle situations when the inputs from the navigation software are not available.

During Ascent In case of RESINS accelerometer failure, navigation will raise a flag. DAP assumes that inertial velocity is valid at the instant of setting accelerometer failure flag. On receiving the accelerometer failure flag, for the first cycle, using the current Inertial velocity nominal phase time is computed from the pre-stored table. Offset between the nominal phase time and the current phase time is also computed, to take care of propulsion dispersions. For the first cycle, computations are carried out using the available inertial velocity. From the next cycle onwards, inertial velocity is computed based on phase time, considering the time offset. With this computed inertial velocity, further computations will continue.

During Descent During the initial phase of the descent, plant dynamics is unstable. So vehicle cannot be flown in open-loop. So previous commands of DAP are retained if RESINS accelerometer fails.

During NzNy control regime in descent, if CSAP alone fails, then Mach based elevon and rudder trim commands are generated from the pre-stored table. If the RESINS accelerometer alone fails, as Mach will not be available, time based elevon and rudder trim commands are generated. If RESINS accelerometer failure follows CSAP failure, there will be switch over from Mach based trim command generation to Time based trim command

generation. To take care of this, offset between the nominal Mach and current Mach is computed. This offset is used in the further time based trim command generation.

Verification and Validation of MMC Software

Peer reviews and independent verification-code inspection and module testing, of each MMC software component were carried out. Integrated MMC software underwent designer-level validations at Simulated Input Profile (SIP) test bed. Inter-processor communication, synchronization, error handling and redundancy aspects were validated extensively at Integrated Processor Test (IPT) bed. Extreme disturbance cases were simulated in on-board processors In Loop Simulation (OILS) test bed. The performance of the flight software was evaluated thoroughly at avionics bay integrated checks. End-to-end signal flow from sensors to actuators was also verified.

Flight Performance

MMCs performed successfully during all the phases of the mission. MMCs internal health remained normal throughout the mission. Relative clock drift in MMC-R was around 1.2 ppm over a design specification of 75 ppm. Execution time margin in MMC matches with pre-flight values. All

the navigation functions were achieved with the required accuracies. All Real-Time Decision (RTD) events were detected as expected and the sequencing commands were issued as per the timing requirements.

Conclusion

The complex mission requirements demanded by RLV-TD HEX-01 vehicle have been captured successfully in the MMC software components. Extensive open-loop and closed-loop simulations were carried out to validate functional and error handling requirements as well as the communication and synchronization aspects.

These efforts have culminated in the successful maiden flight of RLV-TD HEX-01.

References

1. i960 Extended Architecture Programmer's Reference Manual, Intel (1992)
2. A. Burns, A. Wellings, *Real-Time Systems and Their Programming Languages* (Addison-Wesley, Boston, 1990)
3. P.A.S. Laplante, *Real Time Systems Design and Analysis*, 3rd edn. (Wiley, New York, 2005)
4. N. Viswanadham, *Reliability and Fault-Tolerance Issues in Real-Time Systems* (Indian Academy of Sciences, Bengaluru, 1987), pp. 221–230